IMPLEMENTING A GENERIC FRONT PANEL FOR AN SCA RADIO

Serge Harnois (Serge.Harnois@Ultra-TCS.com);
Denis Couillard (Denis.Couillard@Ultra-TCS.com);
Steve Bernier (steve.bernier@crc.gc.ca);

ABSTRACT

One of the key benefits of the SCA specification is that it standardizes how software applications are installed, launched, and controlled. The specification describes a number of programming interfaces used to control software components. Every SCA application can be controlled using the same programming interfaces. Thanks to the standard interfaces, it is possible to implement generic tools to monitor and control SCA-compliant platforms. Such tools run on a personal computer and provide a fine level of control over a radio.

However, in most cases, it is not practical to control a radio via a personal computer. Also, radio operators don't always need the most detailed level of control. This paper explores how a radio user interface can be implemented without the use of a personal computer and still be generic. The paper first explores the SCA programming interfaces that can be used to implement a generic tool for run-time monitoring and control. Then, it provides a discussion on how to design an SCA application such that it can easily be controlled generically. The paper also describes a design approach that allows a radio front panel to be independent of waveform applications. It concludes with a discussion on how simple it is to implement a virtual front panel when a radio is implemented as suggested in the paper.

INTRODUCTION

The Software Communication Architecture (SCA) specification defines several aspect of embedded system design. It defines the cross platform communication infra-structure with CORBA, it defines a variety of components with their characteristics, it defines a set of standards application programming interfaces (APIs) to exchange information between components, and it also defines a useful application deployment mechanism in order to launch and teardown applications.

However, one of the areas where the SCA specification is less precise is the interface to control and monitor applications. This is the subject this paper will attempt to address by identifying which SCA APIs are suitable for this aspect of a radio design. The paper also tries to identify design patterns and best practices that can be applied to facilitate the development of generic radio monitoring and control tools. This paper will also present an example of a specific Human Machine Interface (HMI) implemented for a real platform using those techniques.

1. SCA APIS TO CONTROL AND MONITOR APPLICATIONS

The first step is to explore which SCA APIs can be used to implement a generic tool for run-time monitoring and control. Radio control and monitoring does not impose tight requirements for speed because human reaction time is relatively slow when compared to a computer. Base on these facts an API like *PropertySet* can be used as an interface between any generic or specific tool and SCA applications.

The PropertySet interface provides two functions, one to get a property value called “query” and the second one to change a property value called “configure”. Those functions can work with a several types of data from a simple integer to a complex structure.

The definition of a generic monitoring and control tool here refers to any tool able to recognize SCA APIs without having any knowledge of the specific platform or waveform applications running on the radio. These kinds of tools must use exclusively SCA APIs to exchange information.

The definition of a specific monitoring and control tool as opposition of a generic one is allowed to contain some knowledge of the platform and also the waveform applications. The specific tool will allow a more personalized look and also it could be a lot more intuitive for a user. These kinds of tools could use proprietary APIs to exchange information however the portability of the tool will be affected.

The flexibility of the *PropertySet* interface can easily provides a suitable way to expose an understandable interface for each possible kind of platform and waveform application. This interface allows to configure and query properties of all types, i.e. simples (e.g. integers, strings, and floating points), sequences of simples, structure of simple ,and even sequence of structures. Again, this is not the most efficient way to exchange information in term of speed but it is in term of visibility of your applications features and characteristics.

When applications possess that kind of readable and understandable interfaces, it is significantly easier to implement specific monitoring and control tools, such as HMIs and Web Pages. Properties are described in the SCA Application metadata (XML files). The combination of the name, type, and description allow properties to talk for themselves. A Web page can be developed in no time with properties attached to Web pages dynamic elements. Same thing for an HMI, where the visual interface can be specific to the radio provider equipment, but still rely on properties for the dynamic content.

The following figure shows how simple a Radio monitoring and control tool can be integrated to an SCA design with the *PropertySet* interface. No need to define specific Interface Definition Language (IDL) and develop special code to use it. The entire mechanic is already present in the SCA standard interface part of the Core Framework.

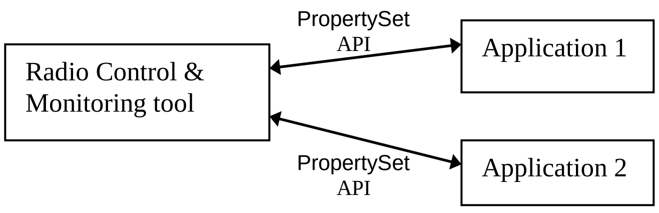

Figure 1 - PropertySet usage for SCA applications control & monitoring

In radio control and monitoring, there is also a need for data refreshing aspect. An HMI is not very useful if the data displayed is out of date or if the user needs to refresh it manually. Another SCA standard API can be used to fulfill this aspect of the tool design; the Event Channel is perfectly adapted for this kind of requirement.

As shown in figure 2, the Multiple Input Multiple Output (MIMO) structure of an event channel allows more than one application to broadcast information intended to be refreshed by any listener on the event channel. It also allows more than one listener, in that case HMIs and generic control and monitoring tools, to receive this information and update their display if needed.

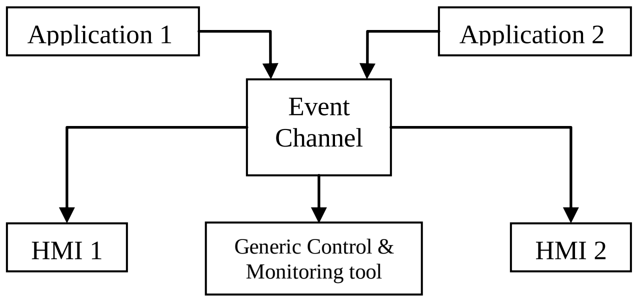

Figure 2 - Event Chanel usage for SCA applications control & monitoring

2. DESIGN PRATICAL APPROACHES HELPING THE DEVELOPMENT OF EXTERNAL TOOLS

In the previous section, APIs have been identified to facilitate the implementation of radio control and monitoring tools. In this section, some design patterns and best practices that can be used at the radio application level will be suggested in order to ease the information exchange between the radio and the control & monitoring tools.

The first concept, a design pattern called proxy, has been briefly introduced in the previous paragraph. This design pattern is used at the property level to publish some platform or application attributes to external applications. This design pattern allows an application developer to advertise a platform feature without having to redesign the feature, simply by using a property at the assembly Controller level. The Assembly Controller property will invoke the *query* operation of the platform component control port to get its property value. The same mechanism is used to modify the property value, this time the *configure* operation will rather be used.

The same concept can be applied for specific APIs, instead of invoking *query* and *configure*, just use the specific operations advertised by the API.

The second concept is a best practice based on the fact that the Assembly Controller of an SCA application is always the only entry point of the application. In that way, if a property located in another resource or device has to be available to an entity outside of the application, a property shall be present on the Assembly controller as a proxy to the other one.

The same concept applies to method on specific ports. Properties at the Assembly Controller need to be implemented as proxies to those methods. The following figure illustrates this concept showing the assembly controller as a gateway to external applications.

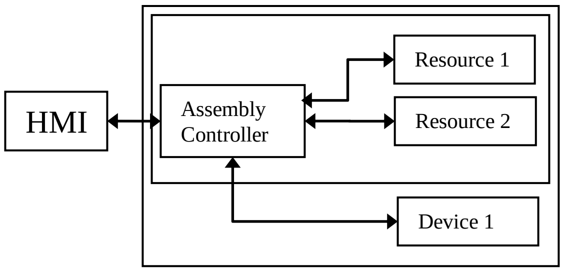

Figure 3 – Assembly Controller as a gateway to external applications.

The last concept is a best practice to help an external control & monitoring tool to update its information displayed to the user. This concept specifies that each property that is public to the application should generate an event with the name of the property and its value each time the property is modified.

That way, any application listening to the event channel can react on those events and refresh its displayed information if necessary.

3. IMPLEMENTATION OF AN HMI BASED ON THE BEST PRACTICES

To illustrate those concepts and demonstrate their real benefits, an HMI named Virtual Front Panel has been developed to monitor and control a real SCA radio, the AN/GRC-245 HCLOS radio from Ultra Electronic-TCS. (Shown in Figure 4) This SCA-based radio is actually in production and deployed in the battlefield. That particular radio was designed in accordance with the best practices described in section 2.

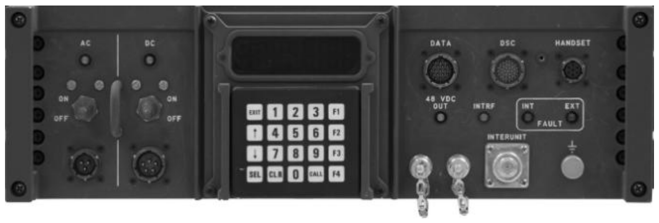

Figure 4 – ULTRA Electronics-TCS AN/GRC-245 HCLOS SCA radio

The Virtual Front Panel needed to react exactly the same way as the physical front panel of the radio and allow the user to navigate through the radio menus and change radio settings.

For instance, when a user changes the radio data rate, a call to the configure function is made on the PropertySet interface to change the waveform application data rate property. (Refer to Figure 5 - step 1)

Then the waveform application goes and configures other internal components to do the necessary changes. (Refer to Figure 5- steps 2 & 3)

When the configuration is completed, the data rate property send an event containing the property name and its new value. (Refer to Figure 5- step 4)

Finally the Virtual Front Panel receives the event and refreshes the display with the new radio data rate.

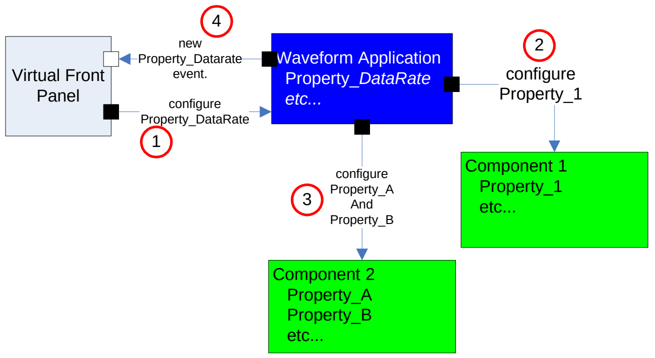

Figure 5 - Property configuration by the Virtual Front Panel example

The radio SCA software architecture was already meeting the majority of the best practices and design pattern described in the section 2 at the exception of the event channel broadcast.

An event channel has been incorporated in the design and in less than a week the radio was producing event on all relevant properties in order to refresh the Virtual Front

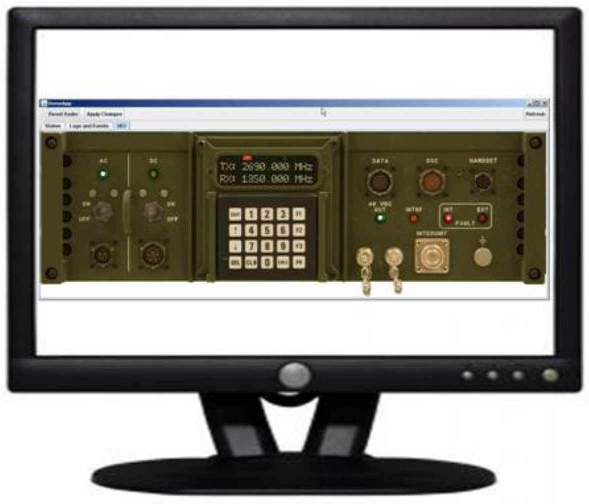

Figure 6 – Virtual Front Panel Application

The Virtual Front Panel resides on a desktop computer using Windows 7 as the OS and it has been developed in Java.(see Figure 6) The radio is a proprietary platform running the INTEGRITY OS and software has been developed in C++.

That demonstrates the benefits of the platform independence provided by the SCA framework and also the power of standard interfaces described in section 1.

This project was a research and development project and it has been realized with a short time period only one month.

The development of the Virtual Front Panel GUI and of the radio infrastructure has been done by two different teams at different locations. A small specification document has been written by the radio design team to describe the overall architecture, along with the specific events and properties involved in this design. With this information, the GUI developer in the other team was able to create test components to emulate the radio events and properties, and to feed the Virtual Front Panel during the development. The time for integration was extraordinary short, with the GUI working on the first day of the integration.

CONCLUSION

The use of the standard SCA API *PropertySet* and the event channel are perfectly adapted to allow control and monitoring tools to interface with SCA applications.

When applied to an SCA radio design early on, some best practices and design patterns can facilitate the development of control and monitoring tools.

The Virtual Front Panel project demonstrates with success the benefits of all the concepts described by this paper.

ACKNOLEDGEMENTS

We would like to express our gratitude to the Communication Research Center Canada and Ultra Electronics – Tactical Communication Systems for supporting us in this project.

REFERENCES

[1] JTRS Standards -Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS)., "SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION", *Version 2.2.2,* May 15, 2006